1995117357

N95- 19767

PalymSys™ - An Extended Version of CLIPS for Construction and Reasoning Using Blackboards

Travis Bryson Dan Ballard
Reticular Systems, Inc.
4715 Viewridge Ave. #200
San Diego, CA 92123

Abstract

This paper describes PalymSys™ -- an extended version of the CLIPS language that is designed to facilitate the implementation of blackboard systems. The paper first describes the general characteristics of blackboards and shows how a control blackboard architecture can be used by AI systems to examine their own behavior and adapt to real-time problem-solving situations by striking a balance between domain and control reasoning. The paper then describes the use of PalymSys™ in the development of a situation assessment subsystem for use aboard Army helicopters. This system performs real-time inferencing about the current battlefield situation using multiple domain blackboards as well as a control blackboard. A description of the control and domain blackboards and their implementation is presented. The paper also describes modifications made to the standard CLIPS 6.02 language in PalymSys™ 2.0. These include: 1) A dynamic Dempster-Shafer belief network whose structure is completely specifiable at run-time in the consequent of a PalymSys™ rule, 2) Extension of the *run* command including a continuous run feature that enables the system to run even when the agenda is empty, and 3) A built-in communications link that uses shared memory to communicate with other independent processes.

Introduction

This paper describes the extensions made to the CLIPS 6.02 language during the design and implementation of a Situation Assessment (SA) expert system for use aboard Army helicopters. An SA system uses data gathered from external environmental sensors, intelligence updates, and pre-mission intelligence to monitor and describe the external environment. An SA system searches for external entities of interest (EEOI), recognizes those EEOIs, and then infers high-level attributes about them. An EEOI is anything that has the potential for affecting the planned rotorcraft mission. EEOIs are primarily (although not necessarily) enemy forces. In order for the system to perform the inferences necessary to develop an assessment of the current situation, it must utilize extensive knowledge about the EEOIs including knowledge about their doctrine, capabilities, probable mission objectives, intentions, plans, and goals. All of these elements combine to form a complete situation description. For a thorough description of the domain problem see [1].

The SA system has been implemented in an extended version of CLIPS called PalymSys™. The SA system implementation makes use of two domain blackboards - current assessment and predicted assessment, as well as a control blackboard for overall control of the system. PalymSys™ provides a reasoning under uncertainty mechanism that handles contradictory and partially contradictory hypotheses and allows multiple hypotheses to coexist. A continuous run option has been added that allows the system to run even when the agenda is empty. Continuous run enables the system to wait for new environmental state data to be provided by the system's sensor subsystems. As new data becomes available, additional reasoning is then performed.

Blackboards

The SA system uses a blackboard architecture as a paradigm for solving the situation assessment problem. The blackboard architecture approach to problem solving has been a popular model for expert system design since the development of the Hearsay-II speech understanding program in the 1970s. It also serves as a framework for the blackboard control architecture - an extension of the blackboard architecture - which is the method of control used in the SA system. The blackboard model for problem solving consists of three primary elements [2, 3]:

Knowledge Sources: The knowledge necessary to solve the problem is partitioned into separate and independent knowledge sources. The independence of knowledge sources means that major modifications to the system should not be necessary when more rules are added to the sys-

PRECEDING PAGE BLANK NOT FILMED

376

tem. In CLIPS and PalymSys™, knowledge sources take the form of rules.

Blackboard Data Structure: A global data structure where knowledge that has been brought to bear on the problem is stored. The blackboard represents the current state of the problem solution. The system attempts to combine and extend partial solutions that span a portion of the blackboard into a complete problem solution. Communication between knowledge sources takes place solely via the blackboard. In CLIPS, a blackboard data structure can be represented by objects that encapsulate the knowledge at each level. The knowledge is contributed by the consequent of rules whose antecedent has been satisfied.

Control: Each of the knowledge sources opportunistically contributes to the overall problem solution. Each knowledge source is responsible for knowing the conditions under which it will be able to contribute to the problem solution. In CLIPS, this means deciding which rule or set of rules should fire next given the current state of the blackboards. Our method for achieving this is the use of the control blackboard architecture. The control blackboard is an extension of the traditional blackboard architecture and will be discussed in detail later in this paper.

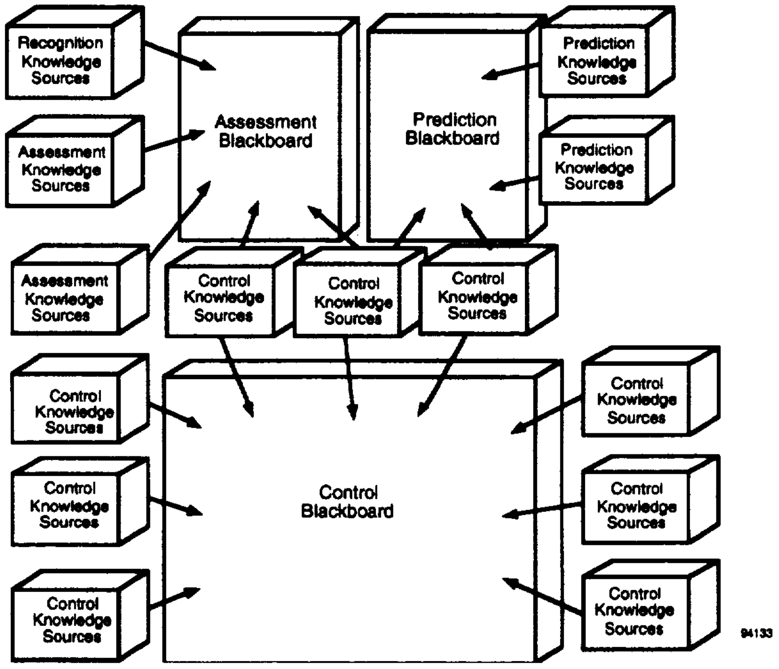
Recognition Knowledge Sources
Assessment Blackboard
Prediction Blackboard
Prediction Knowledge Sources
Assessment Knowledge Sources
Prediction Knowledge Sources
Assessment Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
Control Blackboard
Control Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
Control Knowledge Sources
94133

Figure 1. The Situation Assessment System Architecture

The SA system uses three concurrently executing blackboards for developing a problem solution. These are a *prediction* blackboard, an *assessment* blackboard, and a *control* blackboard. Each blackboard provides storage for the problem solution state data. The assessment blackboard contains the current situation description and is primarily concerned with the current intentions, capabilities, and commitments of EEOIs. The prediction blackboard contains predictions for EEOI behavior and the predicted situation description. The control blackboard contains the knowledge that manages and prioritizes all of the rules and provides for overall control of system problem-solving behavior.

Designing the SA Assessment Blackboard

The blackboard model provides only a general model for problem solving. It falls far short of an engineering specification for actually developing a complete blackboard system in CLIPS. However, this general model does provide significant insight in how to implement complex knowledge-based systems. The first step in designing a blackboard for a given domain problem is to subdivide the problem into discrete subproblems. Each subproblem represents roughly an independent area of expertise. The subproblems are then organized into a hierarchy of levels from least to most abstract. Correctly identifying the problem hierarchically is crucial and will often be the primary factor that determines the effectiveness of the problem-solving system (or whether the problem can be solved at all). Blackboards sometimes have multiple blackboard panels, each with their own set of levels. That is, the solution space can be segmented into semi-independent partitions.

The knowledge sources used by the system are CLIPS rules that have access to the information on the blackboard. Communication and interaction among rules is solely via the blackboard data structure. Even knowledge sources on the same level must share information through the blackboard. Encoded within each knowledge source are the conditions under which it can contribute to the problem solution.

Figure 2 is an illustration of the assessment blackboard in the SA system. The assessment blackboard is divided into a seven-tiered hierarchy. These levels are concerned with developing an environmental state description, characterizing an EEOI, interpreting EEOI plans, roles, and intents and developing a summary description of the overall situation. Each level of the assessment blackboard is a *part-of* hierarchy that represents a portion of the situation assessment solution for a particular EEOI. There is a gradual abstraction of the problem as higher levels on the blackboard are reached. Information (properties) of objects on one level serve as inputs to a set of rules which, in turn, place new information on the same or adjacent levels. During the problem-solving process, more advanced hypotheses and inferences are placed at higher levels of the blackboard.

The blackboard architecture provides a model for the overall problem-solving (inferencing) process. This model is used to structure the problem domain and identifies the knowledge sources needed to solve the problem. While knowledge sources are independent and each contributes to a partial solution, each knowledge source must be designed to fit into the high-level problem-solving blackboard hierarchy created by the system designer.

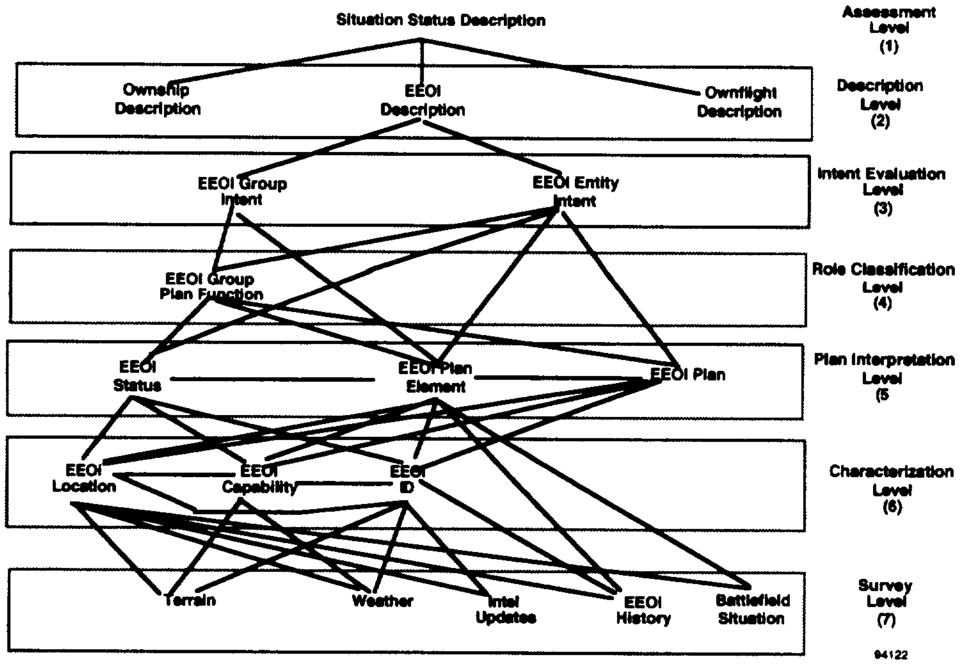

Figure 2. The Assessment Blackboard

Using CLIPS Objects as the Blackboard

Information on the assessment blackboard is represented as CLIPS objects. Figure 3 shows the object representation for knowledge in one of the SA modules. This figure shows the structure in the global plan function (GPF) module at the Role Classification level of the blackboard hierarchy.

A multi-agent plan hypothesis object is created by the system to represent the high-level multi-agent plan of a group of EEOIs where each has the same high-level mission objective. Multi-agent hypotheses and their associated entity plans are stored as objects on the blackboard. An entity plan object contains the sequence of plan elements (activities) that a particular EEOI must actually perform within the context of the associated multi-agent hypothesis. For instance, an EEOI's multi-agent plan might be to capture a refueling depot. A number of EEOIs will be needed to achieve this objective including a security force, a main attack force, and surveillance for the attacking force. The EEOI's entity plan might then be *surveillance* for the attacking force. The multi-agent hypothesis object called *capture refueling depot* encapsulates information local to the role classification level like formation information, hypothesized locations, and typical vehicle types. Other objects can access this information only through the multi-agent hypothesis object's defmessage handlers.

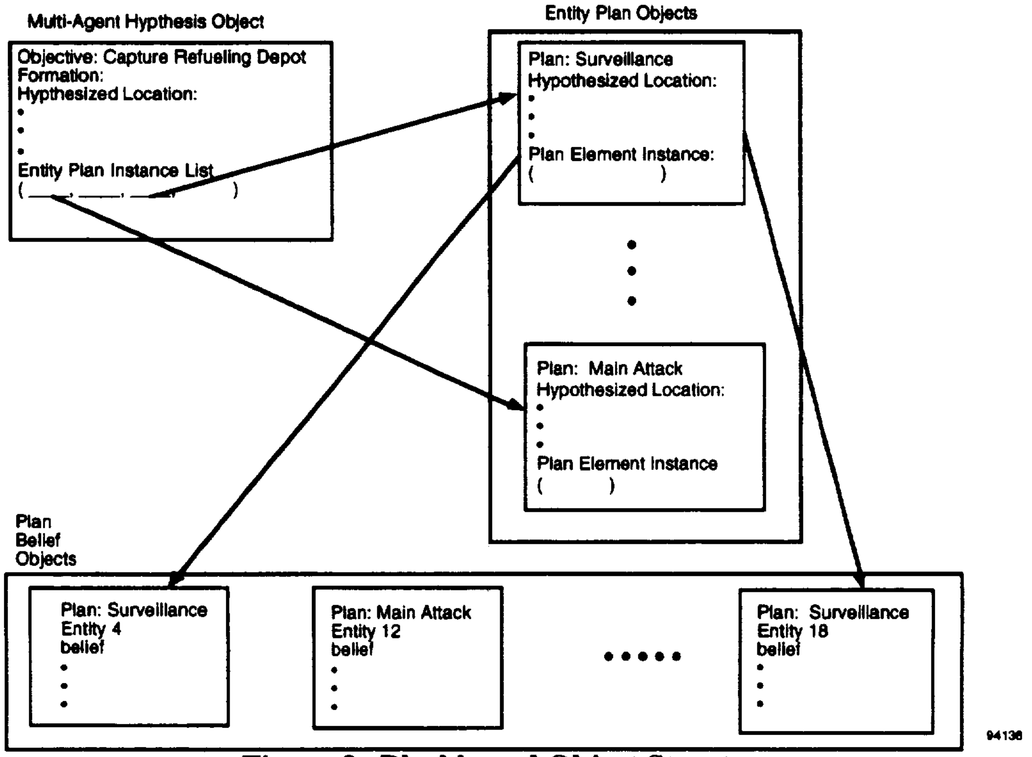

Figure 3. Blackboard Object Structure

The *capture refueling depot* object has a multislot field that contains the list of entity plan instances necessary to carry out its objective. Objects at different blackboard levels which are permanently linked are connected via instance lists. The entity plan instances, in turn, contain the lists of plan elements necessary to carry out the entity-level plan. The plan element lists are encapsulated within the entity plan objects. When the need to do so arises, entity plan objects will search for plan belief objects that correspond to their plan via pattern matching. They search for plan belief objects instead of parsing a pre-defined list because the links in this case are not permanent. The plan belief objects are entity specific and store the degree of belief in which the system believes that a particular EEOI is performing a particular plan. An EEOI may change entity plans or the system may gather evidence that leads it to believe the EEOI is actually performing a different plan. Thus the links between entity plan objects and plan belief objects will change over time.

CLIPS Objects are an ideal data structure for blackboard implementation because they offer encapsulation and easy processing of lists. Recall that the essence of the blackboard approach is that higher levels synthesize the knowledge from the lower levels. The objects at one blackboard level will typically need access to a list of conclusions from the preceding levels. In our approach, objects are always looking down the blackboard, asking other objects for only the information they need. The knowledge at each blackboard level is well encapsulated. Knowledge sources can thus be formulated generically.

The Control Blackboard Architecture

At each point in the problem solving process there are typically a number of knowledge sources that can contribute to the problem solution. Every intelligent system must solve the control problem: i.e., determine which knowledge source should next be brought to bear on the problem solution. In order to solve the control problem it is necessary that control decision making be viewed as a separate problem-solving task. The system must plan problem-solving actions using strategies and heuristics that will help it solve the control problem while balancing efficiency and correctness. The system must become aware of how it solves problems and intelligently guide the problem-solving process.

Explicitly solving the control problem involves providing a body of meta-level (heuristic) knowledge about the domain that is used to guide the control planning process [4]. With such knowledge, the system can reason explicitly about control because the system has access to all of the knowledge that influences control decisions. Meta-level rules then choose domain rules or sets of domain rules that are most appropriate for the current problem-solving situation.

Control knowledge sources interact solely via the control blackboard. The control blackboard is where control knowledge sources post all currently relevant meta-level system knowledge. Partial and complete control plans are stored here. The system also posts high-level control heuristics and problem-solving strategies on the control blackboard.

A well designed control mechanism can make sophisticated meta-level decisions about the problem-solving process. It will seek to make desirable actions more feasible and feasible actions more desirable. It will make plans to seek out important obtainable information when that information is missing. The control mechanism must carefully balance the time spent solving control problems with time spent carrying out domain tasks. It must be aware of how it is solving domain problems and change problem-solving methods to match the situation requirements.

The Control Problem in SA

The SA system is a real-time system that must perform complex inferences within very demanding time constraints. Control is critical in a real-time system because by definition problems must be solved before a deadline. The SA system has a set of meta-level control rules that interact via the control blackboard. The control rule set uses heuristics to evaluate situation characteristics which are used to choose one or more problems from a pool of several competing domain problems. Once the important domain problems are chosen, an efficient control plan is constructed to solve them. A control plan consists of a series of rule groups, or modules, which will be sequentially accessed by the system. A message to work with a specific entity is frequently sent along with the control plan.

Control planning is a way of incorporating efficiency into the system. The meta-level priority criteria do not have to be recalculated every cycle while the system is following an established control plan. The system balances the degree of commitment to the execution of control plans with a variable sensitivity to run-time conditions [5]. The degree of commitment is a function of the uncertainty about the helicopter's actual environment. If the situation is dangerous, the system will lower its commitment to the control plan and heighten its sensitivity to run-time conditions (i.e., incoming sensor data).

Figure 4 is a diagram of the control blackboard used in SA. As in the assessment and prediction blackboards, a multilevel hierarchical blackboard structure is used. The *policy* level is where global focusing decisions are stored. These are heuristically generated at run-time depending on the state of the problem-solving situation. The *problem* level is where domain problems presented to the system are placed. The *strategy* level is where strategies (high-level general descriptions of sequences of actions) are stored. Generating a strategy is the first step in making a plan. Strategy decisions are posted at the strategy level for all accepted problems. A strategy decision is nothing more than a constraint on future actions. The *focus* level is equivalent to the strategy level but is populated by more specific strategies called foci. The *action* level represents the actual sequence of actions chosen by the system to solve the problem [6]. The action level is implemented by the CLIPS agenda mechanism.

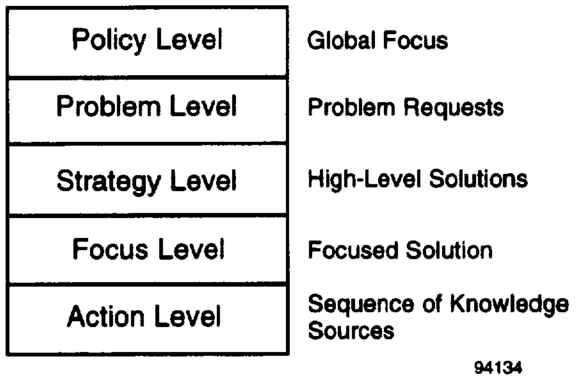

Figure 4. Control Blackboard Hierarchy

Meta Control Plans

The knowledge represented on each of the control blackboard levels increases in abstraction from the bottom level to the top level. However, control is a top-down inferencing process. Unlike the domain blackboards, knowledge at the higher blackboard levels serves as input for the knowledge sources at the lower control blackboard levels. A meta control plan solves the control problem for the control part of the SA system. A meta control plan object is constructed at system start-up that contains the following sequence of phases:

- check for new data
- post foci at policy level
- remove problems (if appropriate)
- request problems
- accept problems
- prioritize problems
- choose problems
- formulate strategy
- formulate plan
- execute a control plan
- record any plan completions and perform system maintenance

The control rules are partitioned by phase. There is a rule set for accepting problems, prioritizing problems, etc. The early phases deal with the higher levels on the control blackboard. The system will cycle through the meta control plan sequentially unless a message is passed from one module to another.

One situation in which message passing occurs is when the system suspects that a new domain problem should be reevaluated in light of the particular domain problem that has been chosen. For instance, when the system is making a plan to do a situation assessment, it will reconsider data that has not yet been integrated into the system. The data may not be intrinsically important. But in the *context* of doing a situation assessment, the system may decide to integrate part of the unprocessed data before doing the situation assessment. Integrating the data first often adds value to the situation assessment because the system will have more information on which to base its assessment. This added value is added to the priority of the data integration plan request. The system goes back to the *prioritize problem* phase and if the data adds enough value to the situation assessment problem, it constructs an object to solve the integrate data problem. The data integration problem instance is added to the list of plan elements in the situation assessment plan object. The actual plan elements to integrate the data into the system are encapsulated within the data integration control plan object.

Heuristics used by the Control Planner

A real-time SA system is continuously supplied with sensor updates, pilot requests for information, anticipation of possible future events, and a plethora of cognitive actions that must be taken in order to assess the current situation. Thus, most of the work of the control part of the system is in deciding what problem to solve. Following are the five domain problems that the SA system solves:

1) Integrate new data into the domain blackboards
2) Focus sensors
3) Generate a current situation assessment
4) Generate a predicted assessment for some opportune time in the future
5) Generate a predicted assessment for time T seconds from the present

In order to solve the control problem, the SA system opportunistically chooses problems from among these five domain problems and makes efficient plans to solve them. This approach provides a built-in well-defined external interface to the system. Problems presented to the SA system by an external agent (e.g., an external system planner or the pilot) are placed in with the problems the SA system has presented to itself at the problem level of the control blackboard.

In order to illustrate the meta-level heuristics that the SA system uses to choose from among competing domain problems, we provide an example of how the SA system integrates new data into the domain blackboard (problem 1 above). The SA system places incoming sensor and intelligence data at the survey level of the assessment blackboard. This new data triggers a problem request at the problem level of the control blackboard. When the SA system decides to integrate the new data, the control rules make and carry out a plan that consists of an ordered sequence of domain modules which will be sequentially examined by the SA system.

All incoming information is rated for importance. EEOIs that have already been encountered by the system are given *Entity Assessment Ratings* (EARs):

EAR = [**Confirmed**(danger), **Plausible**(danger)]

The EAR is a belief function that represents the degree to which an EEOI is a threat to the rotorcraft. The *confirmed* danger is the degree to which the system has confirmed that an EEOI is a danger to ownship. The *plausible* danger (or potential danger) is the worst-case danger that an EEOI presents to ownship at this time. Both of these numbers must lie in the range [0, 1]. The plausible danger is always greater than or equal to the confirmed danger. The less the system knows about an EEOI, the greater the difference between the plausible and the confirmed danger.

The EAR is synthesized into an *Interesting Rating*:

Interesting Rating = 0.7 * Confirmed danger + 1.3 * ability_garner() * (Plausible danger - Confirmed danger)

The system evaluates EEOIs as "more interesting" if there is a large gap between the plausible and confirmed dangers. This means EEOIs that might be dangerous but about which there is little knowledge will be rated or ranked more interesting. Ability_garner is a function that calculates the degree to which the system thinks it currently has the ability to gather more information about the EEOI. When new data come in about an EEOI, we can use that entities' previously calculated Interesting Ratings to prioritize it.

When data about a previously unencountered entity arrives, SA the system favors the integration of the information into the domain blackboard if the data is about a nearby EEOI. The SA system especially favors it if automatic target recognition (ATR) has managed to already confirm an EEOI's vehicle type. Data about new EEOIs is considered intrinsically more important than data about previously encountered entities. The SA system attempts to reject duplicate information before any attempt is made to rate it or integrate it. The control planner always attempts to control the sensors to gain more information about interesting EEOIs.

The amount of time spent generating and evaluating heuristics must be balanced with the amount of time spent executing domain rules. It is possible to expend too many computational resources prioritizing problems and not enough time actively solving them. Entity Assessment Ratings and Interesting Ratings require processing resources for calculation. However, they must be calculated anyway for use by other parts of the system and these calculations are entirely procedural or algorithmic and are therefore computationally relatively inexpensive. Very little extra processing power is required to rate entities in this way. Such overlapping requirements often enable more sophisticated meta-level control knowledge to be produced. The results of the inferencing process represented at various blackboard levels by symbolic abstractions can thus be used as input for procedural/algorithmic computation that, in turn, produces useful metal-level control knowledge.

Using Dynamic Salience for Control

The planning approach to control has the disadvantage of always firing each of the rules that pertain to the chosen domain problem within the modules listed in the control plan. Another layer of control can be attained by directing the system to fire only the subset of eligible domain rules that best apply to the current domain problem. This flexibility is achieved within PalymSys™ by using an expanded form of the salience command.

The modifications to the salience command in PalymSys™ are based on the work done by Robert Orchard of the National Research Council of Canada in his extended version of CLIPS called BB_CLIPS [7]. The added syntax, called rule *features*, are descriptive characteristics of the knowledge contained in a rule that are placed within the antecedent of the rule.

```
;;
;; RULE: pred_pe2
;;
;;     If the EEOI will be able to see ownship and will be able to hit ownship and the EEOI's
;;      plan is combat_recon, main_attack, close_air_support, artillery, or guard then
;;      EE will probably be engaging you in the future (60%).  If not, then we can't be sure
;;      what the EEOI will do next (40%).
;;


(defrule pred_plan_element12
(declare                              ; feature list
       (salience 200)                 ; salience type
       (reliability 35)               ; integer type
       (importance 25)                ; integer type
       (efficiency medium))           ; set type
(Module_focus (focus domain)(sub_focus pred_plan_element)(entity_focus all) (time_focus
?time&:(>= ?time3))(level policy)(BB CONTROL))
(object (is-a EEOI_Pred_location_long) (label ?name) (dist_from_ownship ?dist&:(< ?dist
6))(level interpretation)(BB PREDICTION)) ;; all EEs < 6km away.
(object (is-a EEOI_Pred_capability_long) (label ?name) (see_capability
?seecap&:(>= ?seecap .5)) (hit_capability ?hc&:(> ?hc .5))(level pred_cap)(BB ASSESSMENT))
(object (is-a EEOI_Plan) (label ?name) (propagation ?prop)
        (type ?type&:(member$ ?type (create$ combat_recon long_range_recon guard_forward
guard_flank guard_rear main_attack close_air_support)))(level plan_interp)(BB ASSESSMENT))
=>
        (assert_belief ?name pred_plan_element ?prop ".6 ENGAGE_OWNSHIP .4 ALL")
)
```

Each feature has an associated dynamic salience value determined by its feature arguments. PalymSys™ has a combining function that evaluates the salience of each rule between rule firings. The feature argument itself is a pointer to a dynamic data structure of salience values that is modified by control rules at run-time. For instance, if the system is suddenly faced with a time constraint, a control decision can be made to globally raise the value of the efficiency feature.

A new feature in CLIPS 6.0 is the (set-salience-evaluation every-cycle) command which enables salience values to be calculated dynamically at run-time between rule firings. It is possible to achieve the same functionality described above from within the CLIPS 6.0 shell by placing a function as the argument for the salience command. Between rule firings, the function dynamically computes salience values which are based on global control variables whose values have been determined by control decisions.

A Hybrid PalymSys™/C++ Belief Network

Reasoning under uncertainty is a necessity for a situation assessment system. The SA system must make prescient inferences about such things as an EEOI's plan or the associated elements (steps) in that plan. The EEOI's plans and plan elements cannot be known for certain until various activities are explicitly observed. Other inferences, such as an EEOI's intent or an EEOI's *predicted* plan element, can never be known for certain. Hypotheses must be based on incomplete and unreliable evidence because the battlefield is a complex, uncertain environment. Reasoning under uncertainty requires a probabilistic model of reasoning that supports reasoning using contradictory and partially contradictory hypotheses in which the system has varying degrees of confidence.

PalymSys™ enables the user to construct a Dempster-Shafer (D-S) belief network quite easily. A command line interface allows the user to specify size, structure and number of instances of the network at run time. A belief network propagates the uncertainty associated with a particular piece of knowledge throughout the entire hierarchy of hypotheses that depend upon it. The SA control planner in conjunction with the Rete Pattern Matching algorithm handles the belief propagation through the system hierarchy. When rules are added to the system, no modifications of existing C++ or PalymSys™ code are necessary. By placing a single function call on the consequent of the added rule(s), the system will incorporate the new rule(s) into the belief network automatically. A formal explanation of Dempster-Shafer theory is beyond the scope of this paper. Such detailed presentations can be found in [8, 9, 10]. However, the CLIPS modifications described here can be applied to a monotonic, feed-forward belief network of any type (i.e., Bayesian).

A frame of discernment is a set of mutually exclusive hypotheses. Exactly one hypothesis in a frame of discernment is true at any one time. Each module that uses D-S reasoning in SA has its own frame of discernment. For example, the frame of discernment corresponding to the Plan Element module is the set of fourteen distinct plan elements that an entity is capable of performing within the context of all possible plans. When an entity is encountered, it is assumed to be performing one and exactly one of these plan elements. The purpose of the plan element module is to assign belief values to each of the members of the plan element frame of discernment. A set of propagation values for the plan element frame of discernment is also calculated. The propagation values serve as input to other frames of discernment that use plan elements as evidence.

Recall that the SA system follows entity specific control plans in order to integrate new data into the system. A control plan is an ordered list of modules that the system will sequentially visit to solve a domain problem. The exact order of the control plan will vary depending on what type of data is being integrated into the system. When a control plan element is executed, a particular module fires its rules, assigns belief to the members of its frame of discernment, and then control proceeds to the next module in the control plan which is typically on the next higher level within the domain blackboard hierarchy.

A typical SA domain rule within the belief network will look much like the following rule from the Plan module:

```
RULE: plan_rule12
Description:
;; If the EE's current Plan Element is Reporting then his Plan might be (40%)
;; surveillance, combat recon, or long range recon.  It also might be, to a
;; slightly lesser degree (30%), guard forward, guard flank, or guard rear.  If
;; it's not in those two sets, then the EE could be performing any Plan (30%).
(defrule plan_rule12
(Module_focus (focus domain)(sub_focus plan)(entity_focus ?name)
       (level policy)(BB CONTROL))
(object (is-a EEOI_Plan_element) (label ?name) (type report)
       (propagation ?prop_value&:(> ?prop 0))(level plan_interp)(BB ASSESSMENT))
=>
(assert_belief ?name Plan_Module ?prop_value ".4 SUR CRP LRRP .3 GFR GFL GR .3 ALL")
)
```

The variable *name* is needed as a tag because SA makes an instance of the belief network for each new EEOI encountered. In this case, the assert_belief function places the belief into the frame of discernment associated with the Plan module with a propagation value of *prop_value*. The Module_focus template values in the rule will allow this rule to fire only when the system is firing the rules on the assessment blackboard in the plan module. In this way, the control plan assures that the rules in each frame of discernment for a particular EEOI are finished firing before advancing up to higher levels on the blackboard. The same technique can be used with any monotonic feed-forward blackboard belief propagation scheme.

When evidence is obtained from another frame of discernment, as is the case with the EEOI_plan_element object in the rule above, the propagation value is also sent to the assert_belief function. Evidence without a propagation value associated with it is assumed to have a probability of truth of one. However, the system can easily adapt to any uncertain data by attaching a propagation value to them.

The last rule fired within each module calls the function get_belief to access the appropriate belief and propagation values for the module's frame of discernment. The propagation values will be passed to the next level up in the blackboard hierarchy via the propagation slot in the object that corresponds to the current entity and the current blackboard level.

New domain rules are easily added to the system by placing the assert_belief function on their RHS. No modifications to the reasoning under uncertainty function code are required since propagation is handled entirely by the Rete Pattern Matching algorithm and the SA control planner.

Simulation and Test Environment Interface

SA uses interprocess communication to communicate with our rotorcraft mission simulation and test environment (STE). The STE is a graphical simulation test bed written in C++ and implemented on a RS/6000 workstation. During simulation runs, the STE sends SA sensor information and SA sends the STE directional parameters to control the sensors. A communications class was written for the STE and integrated with PalymSys™. Shared memory in our system is accessed via the standard system C libraries<*sys/shm.h*> and <*sys/types.h*>. More specifically, the STE uses the function calls *shmat*, *shmget*, *shmdt* to attach a process, grab the shared memory and detach the process, respectively.

Standard CLIPS terminates when the agenda is empty. PalymSys™ can be directed to run continuously even though the agenda is empty by adding an optional argument to the run command. The inference engine will idle, waiting for facts to be asserted into the system. This capability is essential whenever the system depends on an independent process, like the STE, as a source for fact assertions.

Three functions were embedded into PalymSys™ in order to communicate with the STE. One function checks the communications link to see if information had been passed over from the STE. The maximum buffer size was 200 characters, so the PalymSys™ function got the name of a file from shared memory that had just been created by the STE. Another PalymSys™ function reads the file just created by the STE and asserts the contents as facts into the PalymSys™ fact base using the *AssertString* CLIPS C library call. Finally, another function lets the STE know via shared memory when SA has sent it information via file transfer. This two-way real-time interprocess communication provides a realistic simulation of a rotorcraft environment.

Summary and Conclusions:

We have implemented a situation assessment blackboard expert system in PalymSys™ -- an extended version of CLIPS. Blackboards are an excellent paradigm for CLIPS expert system implementations. The control blackboard architecture is especially well-suited to real-time applications like SA. We developed a control planner in PalymSys™ that chooses the most important problems to solve based on complex meta-level situation characteristics. The control planner creates domain plans to solve the problems that it chooses. SA uses a monotonic feed-forward Dempster-Shafer belief network implemented in C++. The size and number of instances of the network is dynamic and completely controlled at run-time from the PalymSys™ shell. Finally, we interfaced the SA system to our Simulation and Test Environment using interprocess communication techniques. A continuous run feature was added which enables the inference engine to idle even when the agenda is empty.

Acknowledgments

Mr. Joe Marcelino was instrumental in the implementation of the assessment and prediction blackboards. Mr. Richard Warren implemented the terrain reasoning system. Mr. Jerry Clark provided invaluable advice on the reasoning under uncertainty mechanism used in SA. Mr. Clark served as the rotorcraft domain expert on this project. Mr. Steve Schnetzler implemented the interprocess communications between SA and the Simulation and Test Environment.

Bibliography

[1] D. Ballard and L. Rippy, "A knowledge-based decision aid for enhanced situational awareness," in *Proceedings of Thirteenth Annual Digital Avionics Systems Conference*, Phoenix, AZ, 1994, in press.

[2] H. P. Nii, "Blackboard Systems - Part I," *AI Magazine*, vol. 7, no. pp. 38 - 53, 1986.

[3] H. P. Nii, "Blackboard Systems - Part II," *AI Magazine*, vol. 7, no. 4, pp. 82 - 107, 1986.

[4] N. Carver and V. Lesser, "A planner for the control of problem-solving systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 6, pp. 1519 - 1536, 1993.

[5] B. Hayes-Roth, "Opportunistic control of action in intelligent agents," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 6, pp. 1575 - 1587, 1993.

[6] B. Hayes-Roth, "A blackboard architecture for control," *Artificial Intelligence*, vol. 26, pp. 251 - 321, 1985.

[7] R. Orchard and A. Diaz, "BB_CLIPS: Blackboard extensions to CLIPS," in *Proceedings of First CLIPS Conference*, Houston, Texas, 1990, pp. 581 - 591.

[8] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.

[9] G. Shafer, A Mathematical Theory of Evidence. Princeton, NJ: Princeton University Press, 1976.

[10] G. Shafer and J. Pearl, Uncertain Reasoning. San Mateo, CA: Morgan Kaufmann, 1990.